

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

CONTROLLER FOR AUTOMOBILE

Patent Number: JP7277105
Publication date: 1995-10-24
Inventor(s): KURATA KENICHIRO; others: 04
Applicant(s): HITACHI LTD
Requested Patent: ☐ JP7277105
Application Number: JP19940076845 19940415
Priority Number(s):
IPC Classification: B60R16/02; G05B15/02; G06F9/06
EC Classification:
Equivalents:

Abstract

PURPOSE: To reduce development man-hours for control software and to provide a general purpose applicability in relation to a hardware change.

CONSTITUTION: Automobile control software to be described is constructed of separate components consisting of an application part 1, an I/O description part 2, and an I/O data part 3, and these are connected to each other by means of a software connecting means 4. In this way, description is carried out while the application part 1 and the I/O description part 2 are separated from each other, so that description software is simplified. As the I/O data part 3 is arranged independently of the I/O description part 2, alteration of hardware can be accomplished only by alteration of the I/O data part 3. In addition, an object program is generated by connecting these parts, and therefore the optimization process can be programmed.

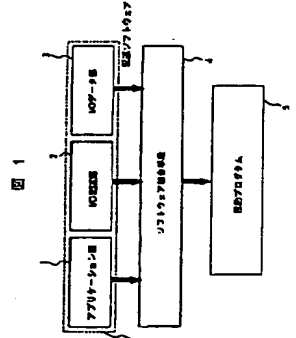
Data supplied from the **esp@cenet** database - l2

(51)IntCl ⁴	発明の分野	特許庁	特許庁
B60R 15/02	M		
G05B 15/02			
G06F 9/06	530 A 7200-5B		
	7531-3H		
	G05B 15/02	P	
	特許請求 請求項の第5 OL (全11頁)		

(21)出願番号	特開平6-78945	(71)出願人	000005108 株式会社日立製作所
(22)出願日	平成6年(1994)4月15日	(72)発明者	東京都千代田区神田區河内町丁目6番地 倉田 隆一郎
		(72)発明者	茨城県日立市大みか町七丁目1番1号 株 式会社日立製作所日立研究所内 吉田 健幸
		(72)発明者	茨城県日立市大みか町七丁目1番1号 株 式会社日立製作所日立研究所内 栗崎 利通
		(72)発明者	茨城県日立市大みか町七丁目1番1号 株 式会社日立製作所日立研究所内 栗崎 利通
(74)代理人	力理士 小川 勝男		

(54)【発明の名称】 自動車用制御装置

(57)【要約】
 【目的】 制御用ソフトウェアの開発工数を低減し、ハードウェアの変更に対して汎用性をもたせる。
 【構成】 記述する自動車制御ソフトウェアをアプリケーション部1、I/O記述部2、I/Oデータ部3に分けた構成とし、ソフトウェア統合手段4によって統合させる。
 【効果】 アプリケーション部1とI/O記述部2を分けて記述するため、記述ソフトウェアが簡便になる。またI/Oデータ部3をI/O記述部2とは独立して設けたのでハードウェアの変更時にI/Oデータ部3の変更だけで済む。さらに、これらを統合して目的プログラム5を生成させるようにしたため、最適化処理がプログラム化できる。



(2) 特開平7-277105

た、ハードウェアへの出力動作記述についても、これら一連のプログラム作成手順の中で同時に検討され、リアルタイム制御がうまく実現されるようプログラム中の好適な位置に、アプリケーションと関連する形で記述され、その記述内容についても、プログラム作成者が使用するハードウェアについてのハードウェアマニュアルを理解し、使用するハードウェアの仕様に合わせて動作手順、内容を記述していた。

[0004]
 【発明が解決しようとする課題】 しかしながら、自動車制御用のソフトウェアを開発するにあたり、このようなプログラムの作成法によれば、プログラム作成時に、プログラムの作成者は本来記述しない制御アプリケーションの内容と共に、使用するハードウェアに合わせた形で、ハードウェアに対する自身の出力処理についても記述しなければならない。したがって、プログラムの作成者は、自動車制御の知識の他に、例えばマイコンのアーキテクチャや、動作法など、ハードウェアに関する多くの知識を必要とした。

[0005] また、自動車の制御に好適なリアルタイム制御を実現するためには、演算、入出力等の処理が最適なタイミング、および手順で行われる必要があり、プログラムの作成にあたっては、これらが全体にわたってうまく動作するように長く考えなくてはならない。したがって、このようなプログラムの作成法によれば、プログラムの作成者はプログラムの全体の流れを把握している必要がなければならぬ。複雑なプログラムの作成時、アプリケーションの変更時には多大な労力を必要とした。特に、複数のプログラムの作成者が共同で作業を行う場合には、この傾向は顕著にあらわれ、作成者全員がプログラム全体の流れを理解しなければならないことになり、きわめて非効率的であると同時に、任意の機能によるミスを誘発するものとなっていた。

[0006] さらに、自動車制御装置のハードウェア条件(コスト、ROM、RAM容量)は厳しく、その上リアルタイム制御であることからその必要速度に対する要求も厳しく、これらすべての条件を満たすようなプログラムを作るためにはプログラムの最適化の作業が欠かせないが、このようなプログラムの作成法ではプログラムの変更のたびにこのような最適化を行う必要があり、作業効率上からいって好ましくない。プログラムの作成者が交代するたびに、あらためてノウハウを会得していかなければならない点もプログラムの生産効率を下げている原因となっていた。

[0007] 本発明の目的は、プログラムの作成時の前記問題を解決し、効率的な開発が可能に、アプリケーション部分の作成に十分な余裕が得られ、かつ全体としては少ない作業工数で、信頼性の高いプログラムの作成が可能となることと、制御アプリケーションの変更やハードウェアの変更などに対して汎用性のある自動

(3) 制御用ソフトウェア構成、およびそれらの結合処理方法を提供するものである。

[0008] 問題を解決するための手段] 前記目的を達成するためには、本発明は、制御用のソフトウェアに、制御内容を記述したアプリケーション部と、ハードウェアに対する入出力処理について記述した1/O記述部とを設け、これらと、1/Oデータ部3に示されたハードウェアに関する情報とを、結合手段を用いて結合し、目的のプログラムを得るようになる。

[0009] (作用) このように構成され、処理される本発明によれば、複雑な自動車制御ソフトウェアの作成時にも、アプリケーション部と1/O記述部2を分けたために、ハードウェアに対する入出力動作に関するソフトウェア設計と、制御のための複雑な動作に関する設計とを別々に行うことができる。さらに、それぞれ別々に設計されたことから、アプリケーションや入出力処理が変更された場合にも記述の一部を修正し、結合することによって、容易に目的のプログラムを得ることができる。

[0010] また、ハードウェアに対する入出力処理内容を1/O記述部2に記述された表現を用いて記述し、ハードウェア自身に関するデータを1/Oデータ部3にそれぞれ分離して記述するようにしたことから、ハードウェアの変更時でも、ハードウェアに付随して1/Oデータ部3を差し替えればその他の部分はそのまま使用する事ができる。

[0011] さらに、別々に作った部品を、結合手段によって結合して目的のプログラムを生成させる方式としたことから、結合手段がプログラムの生成処理のノウハウを記述させることができる。

[実施例] 以下、本発明の実施例を断面に基づき詳細に説明する。

[0013] 図1は本発明の第1実施例である。図において、ソフトウェア開発時に各プログラムの作成者、またはソフトウェア開発者によって記述または改訂される記述ソフトウェアには、アプリケーション部1、1/O記述部2、1/Oデータ部3が設けられている。このアプリケーション部1には、制御対象の動作内容、およびそのための各種を決定するための演算や論理処理などの、制御の本質的な内容が記述される。また、1/O記述部2には前述のアプリケーション部1で必要とされるハードウェアの各種または構成をソフトウェア上に取り込み、あるいはアプリケーション部1において演算、論理処理された結果の制御値や動作内容をハードウェアへ出力する各処理などの、ハードウェアに対する入出力処理、およびその処理タイミングが一般化された表現で記述されている。さらに1/Oデータ部3には、ハードウェアの構成や機能、その設定方法、アドレス

(4) を得る。

[0025] 図6は前記図1で示した第1実施例について、ソフトウェアの結合方法、およびハードウェアへの搭載方法についての一実施例を示したものである。図に示されるように、記述ソフトウェア69にはアプリケーション部61、および1/O記述部62が設けられ、これらは結合手段64によって結合され、生成プログラム65が生成される。また、1/Oデータ63はROM67などに記録された形で提供され、制御装置上66の結合生成プログラム65が必要に応じて呼び出して使用する。

[0026] ただし、生成プログラム65の持つ1/Oデータ63利用機能については、アプリケーション部61、または1/O記述部62にあらかじめ含まれていても良いし、あるいはソフトウェア結合手段65によって結合時に付加されても良い。前記図1から図6において示してきたソフトウェア構成および結合方法を用いることによって、ソフトウェアの記述が簡便になることに加えて、各制御アプリケーションおよび1/O記述をそれぞれ々に設計し、結合することが可能となるので、多数の制御項目をもつ複雑な制御アルゴリズムの記述が容易となり、また、1/Oデータを容易に活用することによって、共通のハードウェアをさまざまな用途に、かつ使いやすいう形で提供することが可能となる。

[0027] 例えばエンジン制御、AT制御のような複雑な制御ソフトウェアも、いくつかの小さなブロックに分けて記述することにより、極めて簡単に制御内容を記述できる。その一例として以下に簡単なエンジンの点火制御モデルの記述を示す。

[0028] 図7は説明するエンジン制御モデルのハードウェアシステム構成を示したものである。

[0029] エンジンの点火は図に示されたプラグ1〜674のギャップ電圧によって行われ、そのため、コイル73に一定時間電圧を渡した後、電圧を切断する制御が行われ、電圧が切断された瞬間、コイルに蓄えられた誘導電流のエネルギーがギャップ間の放電の形で放出される。したがって、コイル73の電流が0a1から0a11するまでの時間（以下過渡時間）の制御は、点火タイミング、すなわち電圧が0a1から0a11するタイミングの制御と共に極めて重要なものである。

[0030] なお、本モデルにおいて、制御用のマイコン7にはエンジンのクランク角の基準位置を知らせるためのRef(図76、クランク角の回転を知らせるためのRef(図77、および吸入空気温度T78、冷却水温度T79などが入力されており、Ref(図76は120度おきに、POS番号77は1度おきに入力される。また、IGN番号77は5度おきに、図77の4つの制御信号が出力されるが、この入力においては、約120度おき点火制御が行われる。実際にはエンジン2の回転に1回だけ点火すればよいので、この出力信号は分

器72によって6つに分配され、6段階エンジンでは、各段階がちょうど720度1回転を完了するようになっている。

[0031] 図8はマイコン86のハードウェア構成を示したもので、本モデルで使用するタイマー89とCPU85が示されている。タイマー89にはタイマカウンタ81や比較器82、コンパレータ83、タイマコンパレータ84があり、タイマカウンタ81には初期値のPOS番号が入り、カウンタが1度回転するたびにカウンタの値も1ずつ増えるようになっている。コンパレータ83の値とタイマカウンタ81の値は常に比較器82によって比較され、2つが一致した時にはタイマ出力87が変化させられ、CPU85に割り込み出力88を発生させることができる。したがってタイマー88を使って点火制御を行うには、タイマーの値を監視するためのタイマコントロールレジスタ84に必要な設定となるように値を書き込むとともに、動作を記した時点のタイマカウンタの値を演算し、CPUからコンパレータ83に書き込む必要がある。

[0032] なお、Re f (値)はCPU85に割り込みの形で入力される。

[0033] 図9は本エンジン点火制御のタイムチャートを示したものである。制御したい内容は、「Re f (値)の立ち上がり97からaだけ経過した時点Y96で1G (値)をoff (出力) する」事と、「コイルの逆起電時にbにする」事であるが、前者は図12に示す「点火時点Y96からaだけ手前の時点Y95で出力をonにする」と記述することになる。

[0034] ここで、前述のコンパレータには動作を記した時点のカウント値を入力しておくことになっているので、たとえば「Re f (値)の立ち上がり97からaだけ経過した時点Y96」をセットするには、まず「Re f (値)の立ち上がり96時点のカウント値S94を読み出す」のではなく、そして、コンパレータに入力する値はs+aである。

[0035] 図10に本モデルを前記図1に示したソフトウェア構成により記述した一実施例を示す。図において、結合前のソフトウェアはアプリケーション値10、1/O配線値102、1/Oデータ値103の3部から構成され、これらは前記図4に示したように、まずアプリケーション値101と1/O配線値102が結合され、ついで、得られたプログラムを1/Oデータ値103を参照しながら展開するという手順で目的のプログラムを得る。

[0036] ここで、各部分の内容について説明する。アプリケーション値10において④に書く一値は、その配線タイミングにおいて、処理を行うことを示す。したがってアプリケーション値101の内容は、まず、Re f (値)が立ち上がった時点 (Re f (値)の立ち上がり時) に、値s+aを

[0037] 図11は前記図10と同様の図面内容で、Re f (値)の立ち上がり時 (Re f (値)の立ち上がり時) に、値s+aをY=s+aと演算Y=s+aを行うことを意味している。つまり④RGJの配線タイミングでX、Yを求める演算を行っているが、④BGJとはバックグラウンド、すなわち「他の処理をしていないとき」、すなわち空いている時間にこの処理を繰り返す行であることを意味している。

[0038] 1/O配線値102に記述される処理は、上から順に、「Re f (値)の立ち上がり時にIGN (点火用の信号) というグループ名のタイマカウンタ値を読み込み、変数Sに代入する」、「Re f (値)の立ち上がり時、IGNというグループ名のタイマーの値がXのとき④aの信号を出力するようにセット処理する」、「IGN出力がonになる時に、IGNというグループ名のタイマーの値がYのとき④bの信号を出力するようにセット処理する」ということを意味している。

[0039] これらも逆起電に行えば、前記図9で示したような目的の制御が実現できる。アプリケーション値101と1/O配線値102の結合においては、基本的には以下の順序にしたがって読み替えを行う。すなわち、同じ配線タイミングで記述された処理については、1/O処理のデータ読み込み、アプリケーションの処理、1/O処理の各種書き込みの順に読み替える。

[0040] このような動作が前記図4で示した結合手段1によって行われるのに対して、1/Oデータ値103を参照しながら、図4における結合手段2による処理が行われる。

[0041] ここで、図12に示した1/Oデータ値103の配線値121、データ122、プログラム展開時の出力123についての情報が記載されている。このプログラム展開時の出力123はソフトウェア結合手段が様々な形で利用できるように対応した形で記述されており、その内容は、ハードウェアのデータ124、構成システムのデータ125、ユーザーの使用形態に合せて記述されるソフトウェアへの引き渡しデータ126、選択的に使用することの多い制御上のデータ127などで、ハードウェアの対応、ソフトウェア開発者の利便性の向上やプログラムの最適化に必要なデータがデータベース化されている。

[0042] このような1/Oデータ値103を参照しながら本点火制御プログラムをCPU85に展開した目的のプログラムを図13に示した。

[0043] 図10において④RGJのついていない処理内容は図13 (c) の④bに記述されており、図13に記述された処理が、結合され、1つのプログラムを形成している。

[0044] ここで、図13 (a) のinitializationに記述された処理内容は前述の図10における表記に直接対応する出力を用いないもので、1/Oデータ値103が結合手段から生成したものである。

[0045] 図11は前記図10と同様の図面内容で、Re f (値)の立ち上がり時 (Re f (値)の立ち上がり時) に、値s+aを

図12に示したソフトウェア構成によって記述したもので、図12の1/Oデータ値103を用い、展開後は前記図13に示した目的のプログラムを得るが、前記図10の場合と比べて、ソフトウェア結合のための処理が1段階減っていることが特徴である。

[0046] 図14は本発明のソフトウェア構成を自動展開ソフトウェア構成に展開した場合のアプリケーション値10、1/Oソフトウェア値102及びアプリケーション値101の3つに分けられる、1/O配線値102及びCPU85 (中央処理装置) に属する1/Oデータ値103をソフトウェア結合手段104に読み込み、作成する。そして、上記1/Oソフトウェア値102及び自動展開OS (図10) をシングルチップマイコン102に展開手段103 (ROM: 読み出し専用メモリ、EPROM: 電気的に書き換え可能なメモリ等) に記憶させ、ユーザーに提供する。ユーザーはアプリケーション値101を作成し、上記配線手段103に書き込み動作を実行する。これにより、例えば、シングルチップマイコン102の性能及びハード構成が変更になった場合、あるいは、1/O構成が変更になった場合でも、ユーザーはアプリケーション値101を移植することができ、アプリケーションソフトをリアルタイムに向上させる。この時、マイコンカーは1/Oデータ値103を変更して自動展開ソフトウェア付きシングルチップマイコン102を提供すればよい。また、プログラムはマイコンのハードを拡張せずアプリケーションソフトを作成することができる。

[0047] 図15はソフトウェア構成を自動展開OS (図10) とした場合の一実施例である。ここでは、上記1/Oソフトウェア値102とタスク管理及び優先順位等から成るタスクディスパッチの基本OS105を用いて自動展開OS106を作成する。このOSは自動車専用であり、自動車制御用に特化した機能もある。また、図14と同様にシングルチップマイコン102の配線手段103に書き込んで提供することも可能である。

[0048] また、目的のプログラムを得るのに、アプリケーション値10と1/O配線値102を分けて記述し、それらを結合する出力を用いないために、アプリケーションや入力処理が変更された場合にも記述の一部を修正し、結合することによって、容易に目的のプログラムを得ることができるので、ソフトウェア利用性が低くなる。さ

らに、ハードウェアに対する入出力処理内容を1/O配線値2に記述した状態を用いて記述し、ハードウェアに関するデータは1/Oデータ値3にそれぞれ分離して記述することから、ハードウェアの変更時でも、ハードウェアに何れも1/Oデータ値3を差し替える。その結果、ハードウェアの汎用性が高くなる。

[0049] さらに、別々に作った部品を、結合手段によって結合して目的のプログラムを生成させる方式とことから、結合手段にプログラムの最適化処理のノウハウを記憶させることができるので、プログラム作成者は特に意識することなしに、プログラム作成技術の継承が行える。

[図面の簡単な説明]

[図1] 本発明による、アプリケーション値、1/O配線値、1/Oデータ値3に分けて記述されたソフトウェアを結合することによって目的のプログラムを得ることを示したソフトウェア作成方法を表す概念図。

[図2] 本発明による、アプリケーション値、1/Oソフトウェア値103に分けて記述されたソフトウェアを結合することによって目的のプログラムを得ることを示したソフトウェア作成方法を表す概念図。

[図3] 本発明による、プログラム値、1/Oデータ値3に分けて記述されたソフトウェアを結合することによって目的のプログラムを得ることを示したソフトウェア作成方法を表す概念図。

[図4] 本発明による、別々に記述されたアプリケーション値、1/O配線値2を結合し、さらに1/Oデータ値3を結合することによって目的のプログラムを得ることを示したソフトウェア作成方法を表す概念図。

[図5] 本発明による、別々に記述された1/O配線値2、1/Oデータ値3を結合し、さらにアプリケーション値10と結合することによって目的のプログラムを得ることを示したソフトウェア作成方法を表す概念図。

[図6] 本発明による、アプリケーション値、1/O配線値2を結合した状態でプログラムを制御装置上に、また1/Oデータ値3をデータ用ROMに書き込み、データを参照しながらプログラムを動作させることを示した構成図。

[図7] エンジンの点火制御を行うに必要なハードウェア構成の説明図。

[図8] タイマーを含むマイコンの動作を説明するための構成図。

[図9] 点火制御モデルの動作を説明するためのタイムチャート。

[図10] 点火制御をアプリケーション値、1/O配線値2、1/Oデータ値3に分けて記述した例を示すプログラム図。

[図11] 点火制御をプログラム値、1/Oデータ値3に分けて記述した例を示すプログラム図。

[図12] 点火制御をアプリケーション値、1/O配線値2、1/Oデータ値3に分けて記述した例を示すプログラム図。

[図13] 点火制御をアプリケーション値、1/O配線値2、1/Oデータ値3に分けて記述した例を示すプログラム図。

[図14] 点火制御をアプリケーション値、1/O配線値2、1/Oデータ値3に分けて記述した例を示すプログラム図。

[図15] 点火制御をアプリケーション値、1/O配線値2、1/Oデータ値3に分けて記述した例を示すプログラム図。

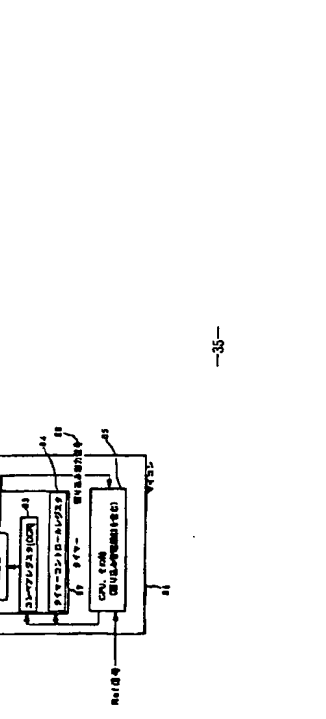
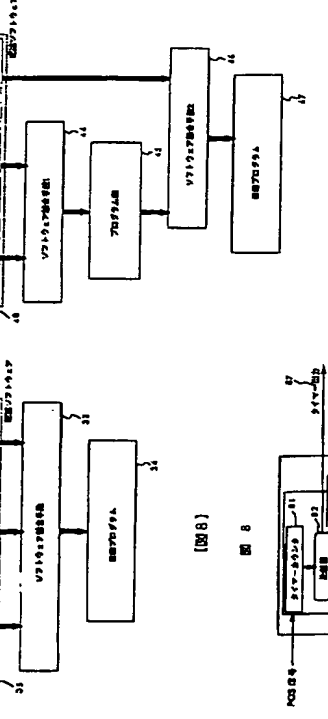
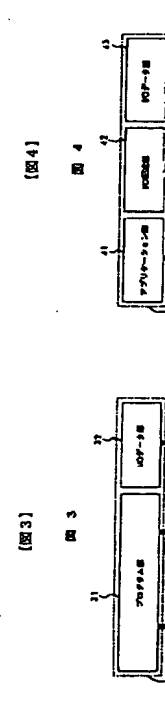
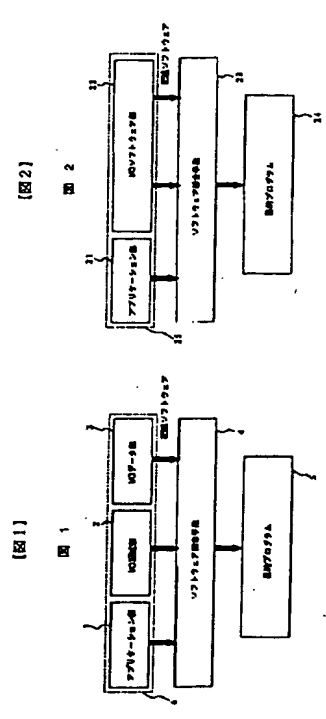
[図16] 点火制御をアプリケーション値、1/O配線値2、1/Oデータ値3に分けて記述した例を示すプログラム図。

[図17] 点火制御をアプリケーション値、1/O配線値2、1/Oデータ値3に分けて記述した例を示すプログラム図。

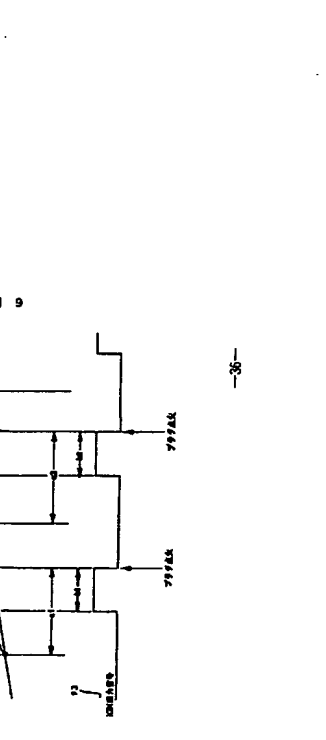
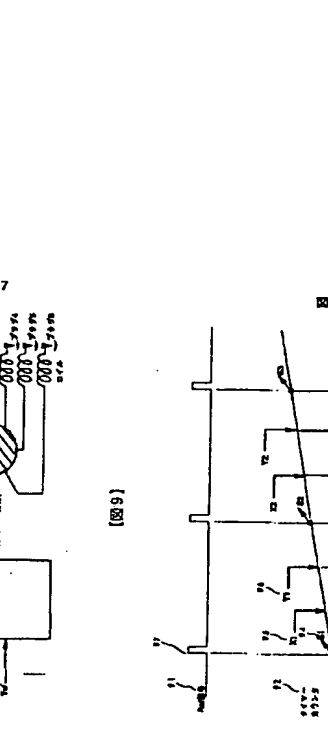
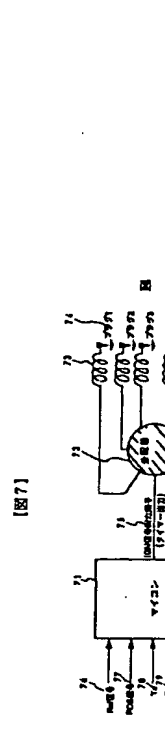
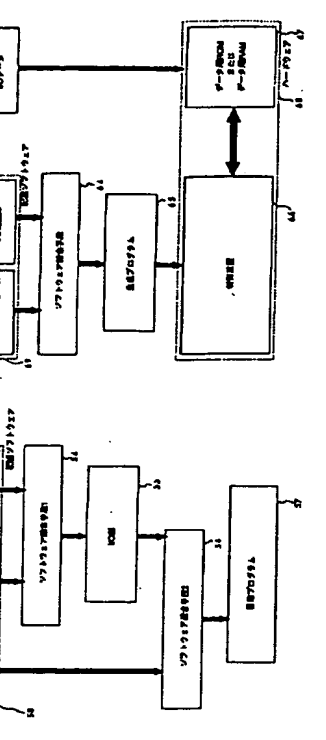
[図18] 点火制御をアプリケーション値、1/O配線値2、1/Oデータ値3に分けて記述した例を示すプログラム図。

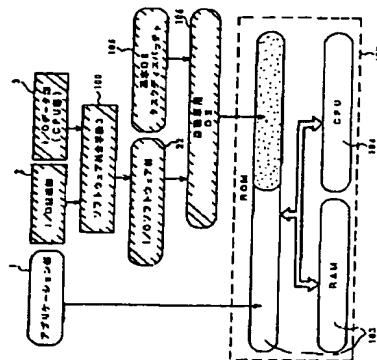
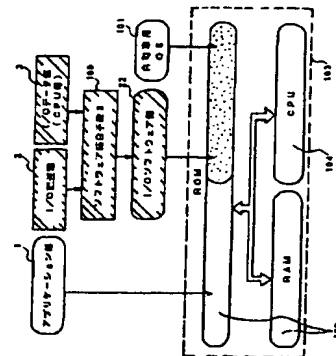
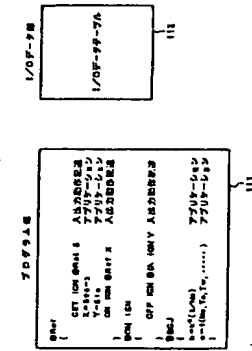
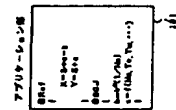
[図19] 点火制御をアプリケーション値、1/O配線値2、1/Oデータ値3に分けて記述した例を示すプログラム図。

【図12】ハードウェアデータを結合手段が利用できる形に整理して示したI/Oデータテーブル図。
 【図13】本発明による分離記述を行ったソフトウェアが、結合処理によって一連のソースプログラムとなることを示したプログラム図。
 【図14】本発明のソフトウェア構成を自動運用シグナルチップマイコンに展開した場合の一実施例を示す図。



【図15】本発明のソフトウェア構成を自動専用OSプログラムとした場合の一実施例を示す図。
 【符号の説明】
 1...アプリケーション部、2...I/O部、3...I/Oデータ部、4...ソフトウェア結合手段、5...目的プログラム。





【圖 12】

12

[illegible]

【図13】

図 13

(a) Initializationへ登録

```

char pldm;
Init_Ign()
{
    IGN_TCR = IGN_TCR_INIT_DATA;
    IGN_TCEN = IGN_TCEN_INIT_DATA;
    SYSCAL = SYSCAL_INIT_DATA;
    SYSCAL = SYSCAL_INIT_DATA;
    PADDR = PADDR_INIT_DATA;
    PADDR = PADDR_INIT_DATA;
    IPRA = IPRA_INIT_DATA;
    IPRA = IPRA_INIT_DATA;
    IPRA = IPRA_INIT_DATA;
}

```

(b) Background Jobへ登録

```

On_Ign()
{
    b = k * (1/Na);
    a = f(No, Pa, Ta, ...);
}

```

(c) Ref Jobへ登録

```

Ref_On_Ign()
{
    register char s = IGN_CMT;
    X = s-a-b;
    Y = s-a;
    #pragma asm
    STC.W SR, 0-SP
    LOC.W $SR16, SR
    #pragma endasm
    if (ID>IGN_CMT)
    {
        IGN_OCRA = X;
        IGN_TCR = IGN_TCEN_ON_DATA;
        IGN_TCEN = IGN_TCEN_ON_DATA;
    }
    #pragma asm
    LOC.W $SR16, SR
    #pragma endasm
}

```

フロントページの続き

(72)発明者 石井 潤市

茨城県日立市大みか町七丁目1番1号 株
式会社日立製作所日立研究所内

(72)発明者 森永 茂樹

茨城県日立市大みか町七丁目1番1号 株
式会社日立製作所日立研究所内